An implementation of Wavelet-based regression under MapReduce

Hanyu Song

December 27, 2016

1 Abstract

Fitting statistical models is computationally challenging for large-scale and high-dimensional datasets. An attractive approach for down-scaling the problem size is to first partition the dataset into subsets and then fit models using distributed algorithms. Motivated by this idea, we propose a novel implementation of wavelet-based linear regression for datasets with longitudinal response under the framework of MapReduce. We provide a theoretical analysis of time complexity of the proposed MapReduce-based algorithm and the conventional approach. Extensive numerical experiments are provided to demonstrate the performance of the new framework. Specifically, we illustrate the possible application of the wavelet-based linear regression in feature selection via implementation under MapReduce.

2 Introduction

In modern science and technology applications, it has become routine to collect complex datasets with large sample size, n, or number of dimensions, p. In such cases, classical feature selection methods may be impractical due to high computational costs. Among these methods, wavelet-based linear regression performs feature selection at each wavelet node via thresholding or shrinking wavelet coefficients. However, this approach, like other classical feature selection, witnesses computational bottlenecks when the dataset is large. In response to this, we focus on datasets with longitudinal response and study the analogue of this algorithm under the framework of MapReduce.

This algorithm constitutes two-stage sequential MapReduce, with the first stage being wavelet transform and the second being linear regression. In particular, in the wavelet transform stage, the key-value pair is the row index of the response matrix and the row of the response matrix $\mathbf{Y}$. In the linear regression stage, the key-value pair is the wavelet node index (or the column index of wavelet coefficient matrix $\tilde{\mathbf{Y}}$) and the column of $\tilde{\mathbf{Y}}$, respectively. In each stage the Reducers only serve to aggregate the output from the Map tasks by key, thereby resulting in zero communication cost between the Mappers and the Reducers.

The rest of the paper is organized as follows. In Section 3, we detail the proposed MapReduce-based algorithm. Section 4 provides the time complexity analysis of this proposed framework and compares its performance to the conventional approach. Section 5 presents extensive simulation studies to illustrate the performance of our framework.

3 MapReduce-based algorithm

3.1 Wavelet-based linear regression

Throughout this paper, we denote the sample size by n, the number of variables by p, and the number of time measurements by T. Suppose we have an n by p design matrix $\mathbf{X}$ and an n by T longitudinal response matrix $\mathbf{Y}$. In the response matrix $\mathbf{Y}$, rows represent individual observations and columns represent time measurements. For instance, for each individual we observe blood pressure measurements from T time points. Taking such T measurements for n individuals gives an $n \times T$ response matrix $\mathbf{Y}$. Denote the column and row vectors of Y by $\{\mathbf{y}_j\}, j = 1, 2, \ldots, T$ and $\{\mathbf{y}^{(i)}\}, i = 1, 2, \ldots, n$ respectively.

To conduct wavelet-based feature selection, we first perform discrete wavelet transform (DWT) on each $\{\mathbf{y}^{(i)}\}$ using any predefined wavelet basis. This procedure yields a set of T-dimensional wavelet coefficients $\{\tilde{\mathbf{y}}^{(i)}\}$. Combining $\{\tilde{\mathbf{y}}^{(i)}\}$ horizontally by row index i gives a wavelet coefficients matrix $\tilde{\mathbf{Y}}$ of the original response $\mathbf{Y}$. Because basic DWT requires the length of the input vector to be a power of 2, we assume T is the power of 2 throughout this paper for simplicity.

Denote the set of n-dimensional column vectors of $\tilde{\mathbf{Y}}$ by $\{\tilde{\mathbf{y}}_j\}, j = 1, 2, \ldots, T$. We simultaneously fit column-wise linear regressions to $\tilde{\mathbf{Y}}$ and $\mathbf{X}$. Specifically, assume the linear model $\tilde{\mathbf{y}}_j = \mathbf{X}\beta_j + \epsilon_j$ for each j, where $\epsilon_j \sim \mathcal{N}(0, \sigma^2\mathbf{I})$. Fitting such linear regression to $\{(\tilde{\mathbf{y}}_j, \mathbf{X})\}$ yields a set of p-dimensional ordinary least square (OLS) estimates $\{\hat{\beta}_j\}, j = 1, 2, \ldots, T$. The estimates $\{\hat{\beta}_j\}$ are indeed the estimated signal strength at each wavelet node. Based on such estimates, we threshold the insignificant estimates at each wavelet node and perform inverse wavelet transform to obtain estimated coefficients at the original scale. Such a procedure constitutes feature selection. To summarise, the above algorithm can be specified as follows:

1. Perform row-wise DWT on the response matrix $\mathbf{Y}$ and save the output in the original row order as $\tilde{\mathbf{Y}}$.
2. Fit T linear regression models to the set of column vectors of $\tilde{\mathbf{Y}}$ and $\mathbf{X}$, namely to $\{(\tilde{\mathbf{y}}_j, X)\}$, $j = 1, 2, \ldots, T$ and save the set of OLS estimates $\{\hat{\beta}_j\}$.

3.2 Wavelet-based linear regression under MapReduce framework

In view of the parallel nature of the above algorithm, we propose its analogue implemented under the framework of MapReduce. The new algorithm consists of two stages of MapReduce, with the key being the row index of the response $\mathbf{Y}$ in the first stage and the column index of the wavelet coefficients $\tilde{\mathbf{Y}}$ of $\mathbf{Y}$ in the second. The new framework can be fully specified as follows:

3.2.1 First-stage MapReduce (Key: response row indices, Value: row vectors of response)

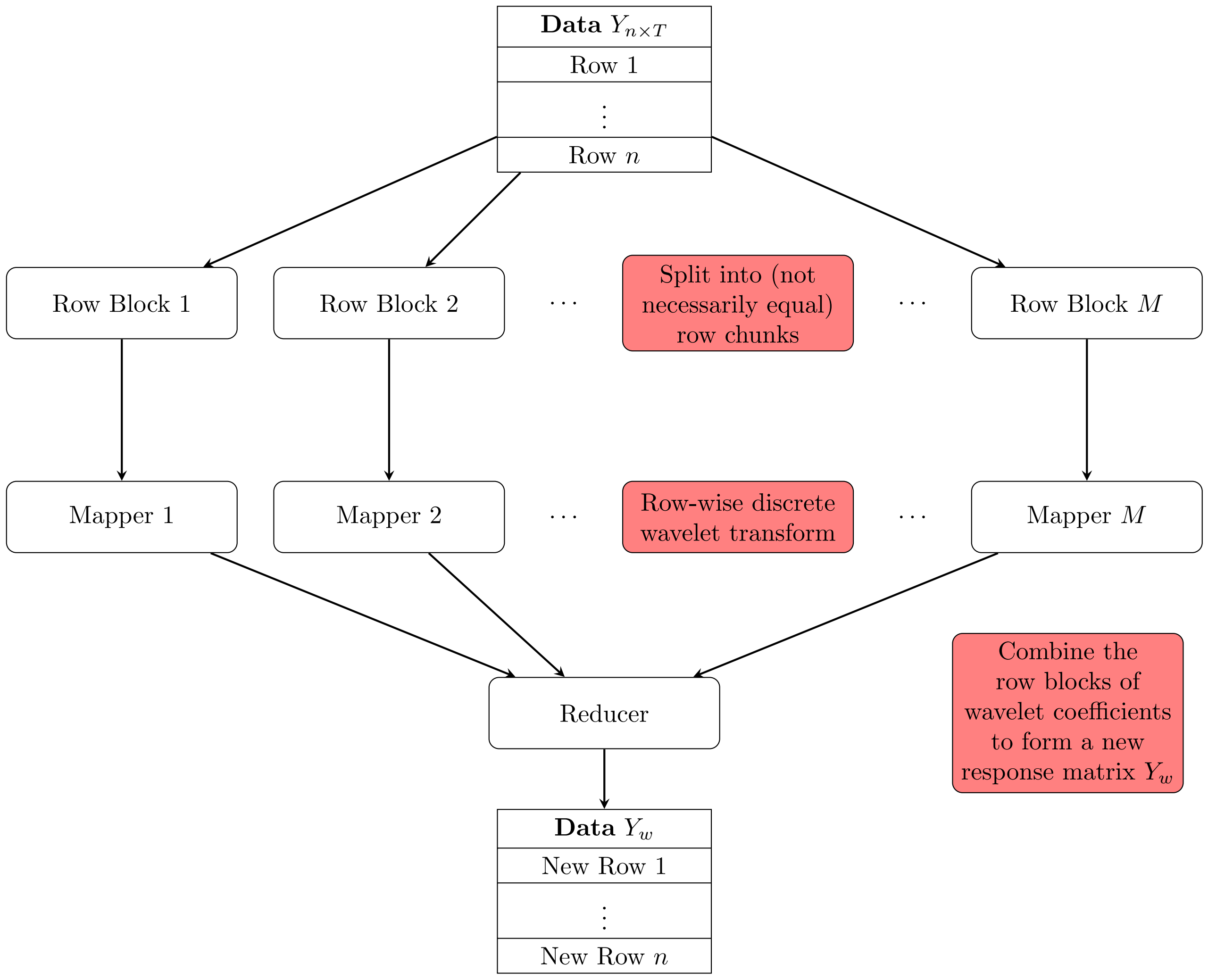
1. Map
   (a) Input the response matrix $\mathbf{Y}$,
   (b) Partition $\mathbf{Y}$ horizontally (in the sample space) into $\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_M$ and distribute to M workers,
   (c) On each subset $\mathbf{Y}_j$ within an worker, perform (row-wise) DWT by key using predefined wavelet basis.
2. Reduce
   (a) Collect the results from M workers and organise by key,
   (b) Output a transformed response matrix $\tilde{\mathbf{Y}}$.

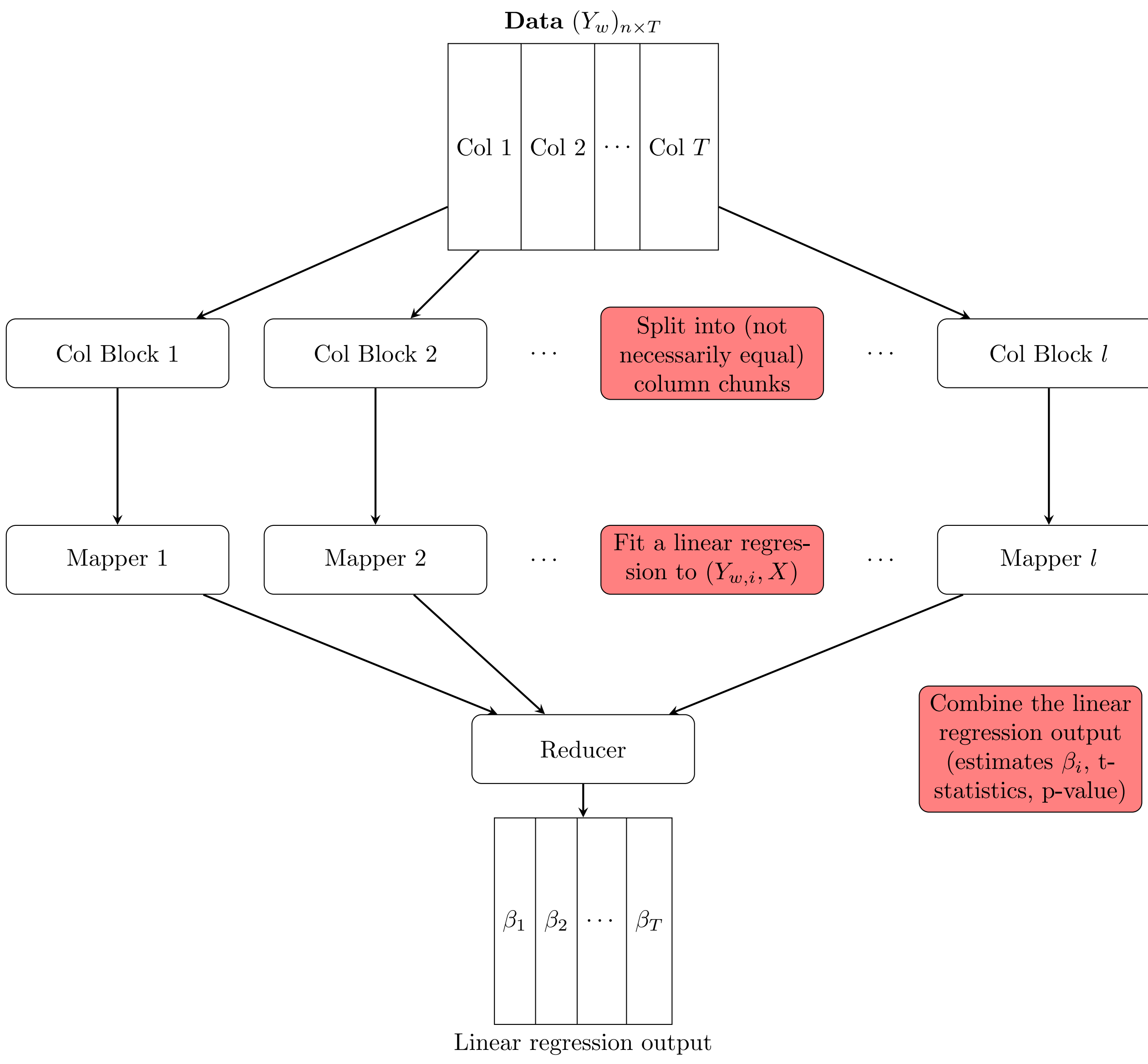
3.2.2 Second-stage MapReduce (Key: column indices of transformed response, Value: column vectors of transformed response)

1. Map
   (a) Input the transformed response $\tilde{\mathbf{Y}}$ and explanatory variables $\mathbf{X}$.
   (b) Partition $\tilde{\mathbf{Y}}$ vertically (in the time space) into $\tilde{\mathbf{Y}}_1, \tilde{\mathbf{Y}}_2, \ldots, \tilde{\mathbf{Y}}_M$ and distribute to M workers.

   (c) On each subset $\tilde{Y}_j$ within an worker, fit a linear regression to $(\tilde{Y}_j, X)$. Specifically, $\tilde{Y}_j = X\beta + \epsilon$, where $\epsilon \sim \mathrm{N}(0, \sigma^2\mathbf{I})$.
   (d) In each worker, output OLS estimates $\hat{\beta}$, standard errors, t statistics and p-values.

2. Reduce
   (a) Collect the results from M workers and organise by key.
   (b) Output fitted results in a table, including estimated coefficients $\hat{\beta}$, standard errors, t statistics and p-values.


Data $Y_{n\times T}$
Row 1
⋮
Row n
Row Block 1
Row Block 2
⋯
Split into (not necessarily equal) row chunks
⋯
Row Block M
Mapper 1
Mapper 2
⋯
Row-wise discrete wavelet transform
⋯
Mapper M
Reducer
Combine the row blocks of wavelet coefficients to form a new response matrix Y_w
Data Y_w
New Row 1
⋮
New Row n


Data $(Y_w)_{n\times T}$
Col 1
Col 2
$\cdots$
Col T
Col Block 1
Col Block 2
Split into (not necessarily equal) column chunks
Col Block l
Mapper 1
Mapper 2
Fit a linear regression to $(Y_{w,i}, X)$
Mapper l
Reducer
Combine the linear regression output (estimates β_i, t-statistics, p-value)
β_1
β_2
β_T
Linear regression output

4 Time Complexity Analysis

In this section we analyze the time complexity of the conventional algorithm and the proposed MapReduce-based framework based on the description in Section 3 and make comparisons. Same as the previous sections, let us denote the sample size by n, number of features by p, and number of time measurements by T.

4.1 Complexity of the conventional algorithm

Step 1 as described in Section 3.1 involves DWT of response matrix $\mathbf{Y}$. Assuming we use Haar wavelet basis implemented via Mallat's (1989) Fast Wavelet Transform (FWT), the time complexity is $O(T)$. In step 2 we need calculate OLS estimates $\hat{\beta}_j$, where $\hat{\beta}_j = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}_j$ for $j = 1, 2, \ldots, T$. Indeed, this is where the computational bottleneck lies. To find $\hat{\beta}_j$, an unavoidable procedure is calculating $(\mathbf{X}^T\mathbf{X})^{-1}$, a step involving matrix multiplication $(\mathbf{X}^T\mathbf{X})$ and matrix inversion with complexity $O(np^2)$ and $O(p^3)$ respectively. In the next step, we multiply $\mathbf{X}^T$ and $\mathbf{Y}_j$ to ease the computational burden to obtain an intermediate p-dimensional vector $\mathbf{z}_j$ in $O(np)$ time. The last step is the multiplication between $(\mathbf{X}^T\mathbf{X})^{-1}$ and $\mathbf{X}^T\mathbf{Y}_j$, which has time complexity $O(p^2)$. Therefore, the total computation cost of calculating OLS is $O(np^2+p^3)$ for each regression and $O(np^2+p^3)$ for the entire dataset. Combining Step 1 and 2 the algorithm has complexity $O(Tn + Tnp^2 + Tp^3)$ for the entire dataset.

4.2 Complexity of MapReduce-based algorithm

Let M be the total number of mappers and R be the total number of reducers. Suppose the size of the response matrix $\mathbf{Y}$, design matrix $\mathbf{X}$ and the transformed response matrix $\tilde{\mathbf{Y}}$ is $|y|$, $|x|$ and $|\tilde{y}|$ respectively, measured in bytes.

In the first-stage MapReduce, key-value pair is (row index i of the response, T-dimensional row vectors $\mathbf{y}^{(i)}$). Same as the analysis in Section 4.1, suppose we use Haar wavelet basis implemented via Mallat's algorithm. Since the time complexity of FWT is $O(T)$, the total Map cost is $O(nT)$ and per Mapper cost is $O(nT/M)$. Because the Reduce tasks only involve aggregating the outputs from the Map tasks, the total Reduce cost scales linearly with the number of keys or, in other words, the sample size n and thus per Reducer cost is $O(n/R)$.

In addition to the computation cost, we are also interested in the communication cost as another quality indicator of algorithms implemented on a computing cluster. By definition, the communication cost of a task is the size of the input to the task measured in bytes. It is important, as many applications witness the bottleneck of moving data among tasks, such as transporting the outputs of Map tasks to their proper Reduce tasks. In our first-stage MapReduce, the input of the Map tasks is the size of $\mathbf{Y}$ and thus incurs communication cost $O(|y|)$. The sum of the outputs of the Map tasks—the size of the transformed response matrix $\tilde{\mathbf{Y}}$—is no larger than their inputs, $O(y)$, since DWTs introduce sparsity to the wavelet coefficients. In addition, given that key—the row index i—is unique and exists only on one worker and each output key-value pair is sent to exactly one Reduce task, it is likely that communication from Map to Reduce tasks occurs memory-to-disk within the worker, rather than across the interconnect of the cluster. Thus, no significant communication cost is incurred from Map tasks to Reduce tasks. Hence, the communication cost of the first-stage MapReduce is $O(|y|)$.

In the second-stage MapReduce, key-value pair is (column index j of the wavelet coefficient matrix $\tilde{\mathbf{Y}}, \tilde{\mathbf{Y}}_j$). In the Map tasks, the computationally intensive part lies in finding the OLS estimates $\hat{\beta}_j$, where $\hat{\beta}_j = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}_j$. Fortunately, $(\mathbf{X}^T\mathbf{X})^{-1}$ is the same across all Mappers, thus no repetitive calculation is required and the one-time computation is $O(np^2 + p^3)$. The remaining computational task for a new $\mathbf{Y}_j$ involves two matrix products, the multiplication between $\mathbf{X}^T$ and $\mathbf{Y}_j$ and the multiplication between $(\mathbf{X}^T\mathbf{X})^{-1}$ and $\mathbf{X}^T\mathbf{Y}_j$, which have time complexity $O(np)$ and $O(p^2)$ respectively. Summing up the computation cost from the one-time calculation of $(\mathbf{X}^T\mathbf{X})^{-1}$ and the remaining repetitive tasks yield the total Map cost $O(Mnp^2+Mp^3+Tnp+Tp^2)$ and per Mapper cost $O(np^2+p^3+Tnp/M+Tp^2/M)$. The Reduce tasks again only involve grouping the result from the Map tasks, indicating the total Reduce cost scales linearly with the number of keys, which is here the total number of wavelet nodes T. Thus, per Reducer cost is $O(T/R)$, minimal in comparison to the per Mapper cost.

Furthermore, the communication cost of the second-stage MapReduce is $O(|\tilde{y}| + |x|)$. To see this, note that the input of the Map tasks is the size of $\tilde{\mathbf{Y}}$ and $\mathbf{X}$. In addition, communication from Map to Reduce tasks occurs memory-to-disk only, due to the same reason explained in the first-stage MapReduce, thereby resulting in negligible communication cost for this procedure. Therefore, the total communication cost, which only stems from the input of the Map tasks, is $O(|\tilde{y}| + |x|)$.

To conclude, our proposed new algorithm implemented under MapReduce is $O(Tn/M + np^2 + p^3 + (n + T)/R + Tnp/M + Tp^2/M)$, a result from summing up all the computation cost per worker. Additionally, the total communication is also minimal, incurring only the size of the input to the Map tasks. Based on this result and comparison with the conventional algorithm in $O(Tn + Tnp^2 + Tp^3)$, we conclude these two algorithms are asymptotically equivalent in the sample size N and number of time measurements T. However given an abundance of Mappers and Reducers, the proposed new algorithm is computationally superior to the conventional one when the number of time measurements is large. Indeed, the relationship remains unclear when a small number of time measurements are observed, since the communication cost may dominate the computation cost. Please see the simulation results in Section 5 for further discussion.

5 Illustrative simulation

In this section, we use synthetic datasets to illustrate the empirical performance of wavelet-based linear regression under MapReduce framework via extensive numerical experiments. To ensure reliable comparison, we verify without illustrating here that the wavelet-based linear regression results are consistent under the proposed MapReduce framework and the conventional algorithm. With this guarantee, we first verify our claim in Section 4.2 that the new algorithm achieves computational gain when a large number of time measurements is observed and other computational properties. Second, we demonstrate an application of feature selection via thresholding or shrinking $\hat{\beta}_j$, $j = 1, 2, \ldots, T$ at each wavelet node.

The synthetic datasets are simulated with $\mathbf{X} \sim \mathrm{N}(0, \mathbf{\Sigma})$ and $\epsilon \sim \mathrm{N}(0, \sigma^2)$. The support of β is $S = \{1, 2, 3, 4, 5\}$. We generate X_i from a standard multivariate normal distribution with independent components. The coefficients are specified as $\beta_{ij} = (-1)^{Ber(0.5)} \left(|\mathrm{N}(0,1)| + 5\sqrt{\log p/n}\right) \mathbb{I}(i \in S)$, where β_{ij} is the coefficient associated with $\mathbf{X_i}$ and $\mathbf{Y}_j$. The variance σ^2 is chosen such that $\hat{R}^2 = var(\mathbf{X}\beta_j)/var(Y_j) = 0.9$, for $j = 1, 2, \ldots, T$. For evaluation purposes, we consider two different structures of $\mathbf{Y}$ as follows:
Model (i) $\mathbf{Y}_j = \mathbf{X}\beta_j + \epsilon$, $j = 1, 2, \ldots, T$.
Model (ii) $\mathbf{Y}_j = \mathbf{X}\beta_j + \eta_j + \epsilon$, $j = 1, 2, \ldots, T$. Let η be the matrix with column vectors $\{\eta_j\}$. Let $\{\eta^{(i)}\}, i = 1, 2, \ldots, n$ be the set of T-dimensional row vectors of η. For each $\eta^{(i)}$, $\eta^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Omega})$, where $\mathbf{\Omega}$ is an n by n matrix with diagonal elements 1 and off-diagonal elements 0.5.

5.1 Comparison: computational performance

As discussed in Section 4.2, the proposed MapReduced-based algorithm scales better in the number of time measurements in comparison to the conventional one. We verify this property by using Model (i). The model dimension and the sample size are fixed at $n = 1000$ and $p = 20$ respectively, and the number of true signals is fixed at $s = 5$. For each model, we simulate 10 synthetic datasets and record the trimmed mean computational time using the middle 50%.

Table 1 shows that when the number of time measurements $T < 512 = 2^8$, the MapReduce algorithm has a higher computation cost that hinders performance in small data sets. However, the MapReduce scales much better than the conventional algorithm, eventually surpassing it.

Table 1: Time in seconds for MapReduce and conventional algorithm to run with T time measurements. Here $n = 1000$ and $p = 20$ for all trials. Reported results are trimmed mean using five out of a total of ten replications.

T	MapReduce	Conventional
8	22.4	1.9
16	28.5	2.8
32	33.9	4.2
64	39.2	6.9
128	48.6	12.0
256	62.0	22.3
512	84.4	48.7
1024	119.4	89.5
2048	136.5	167.6
4096	157.8	324.6

5.2 Performance of applications in feature selection

As discussed before, wavelet-based linear regression can be applied to feature selection via thresholding wavelet coefficients. Here we present the feature selection results implemented with MapReduce. The model dimension, the sample size and number of time measurements are fixed at $n = 1000, p = 10$, and $T = 16$. Both model (i) and model (ii) yield variable selection result consistent with the true model. In particular, I let the first five predictors (1 - 5) to be true signals and the remaining five ones (6 - 10) be noises. Denote the OLS estimates on the original scale, the OLS estimates on the original scale after shrinkage and true coefficients by $\hat{\beta}, \hat{\beta}_{\text{shrink}}$ and β respectively. Below I present the before and after shrinkage multi-resolution plots associated with predictor 1, 2, 6 and 8 for illustration purposes. I also show the $\hat{\beta}$ and $\hat{\beta}_{\text{shrink}}$ to demonstrate the shrinkage in the OLS estimates. From the multi-resolution plots, we observe that non-signal explanatory variables tend to witness more shrinkage at each wavelet node.

5.2.1 Results of simulations from Model (i)

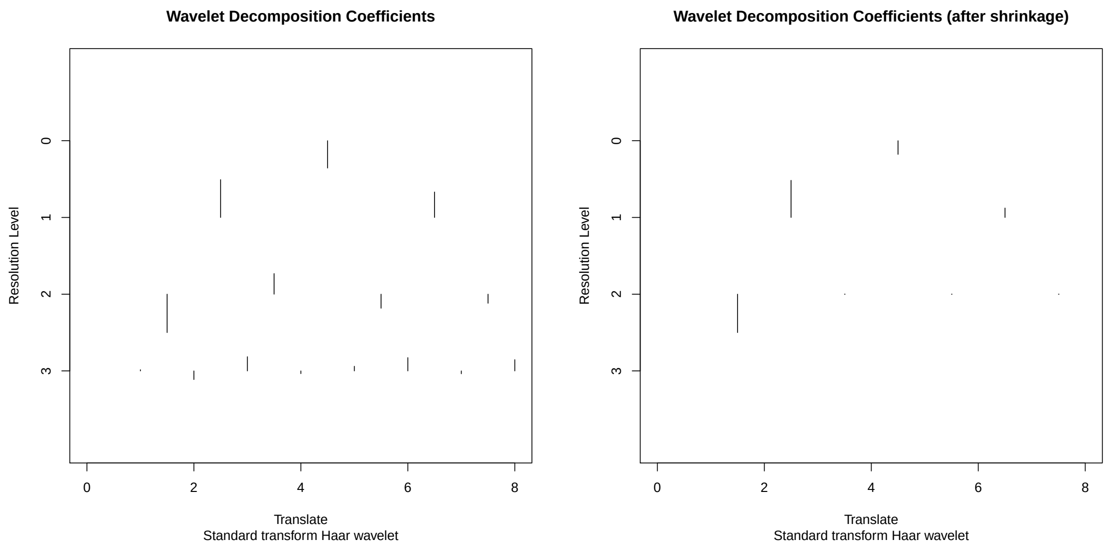


Figure 1: Model (i): Before and after shrinkage multi-resolution plots of $\hat{\beta}_1$

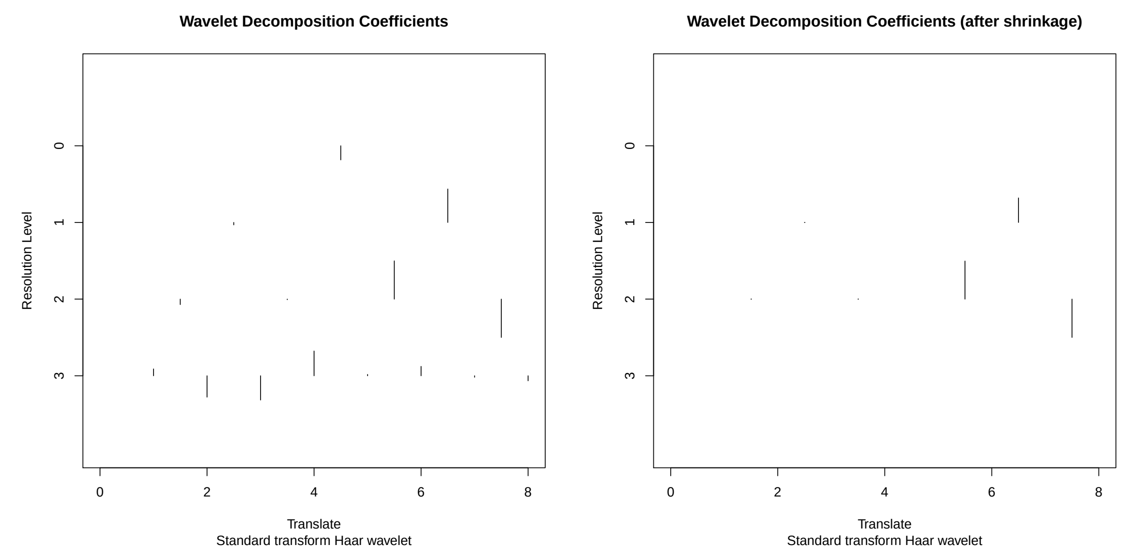


Figure 2: Model (i):Before and after shrinkage multi-resolution plots of $\hat{\beta}_2$

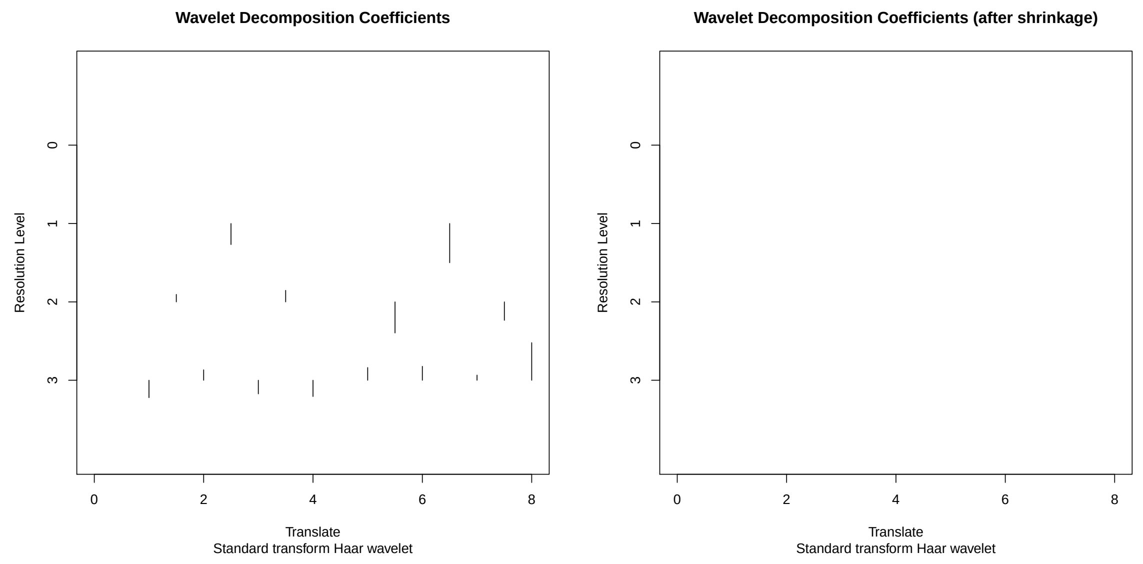


Figure 3: Model (i):Before and after shrinkage multi-resolution plots of $\hat{\beta}_6$

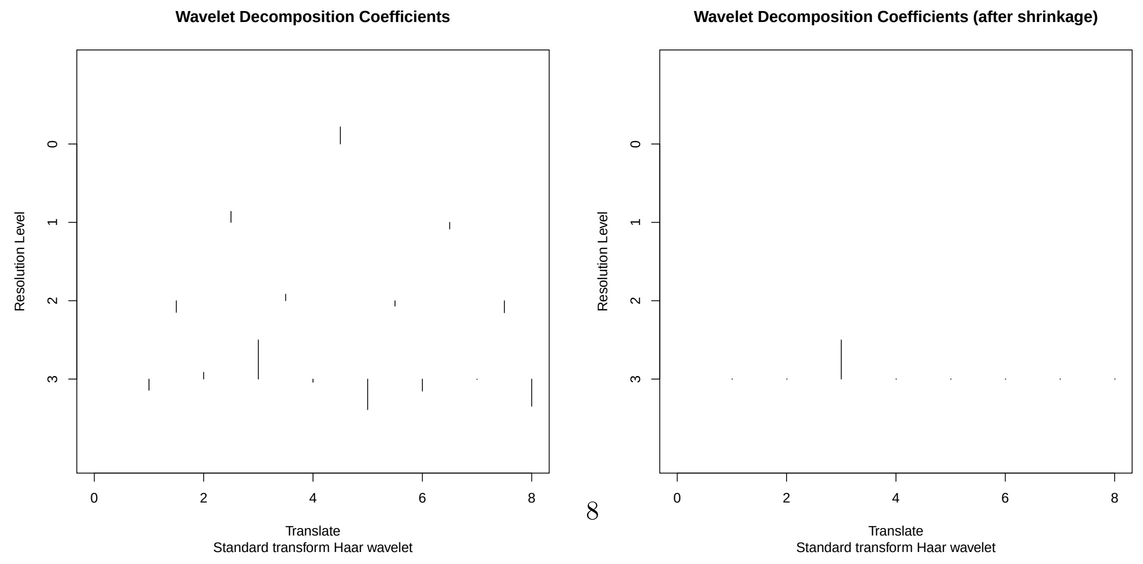


Figure 4: Model (i):Before and after shrinkage multi-resolution plots of $\hat{\beta}_8$

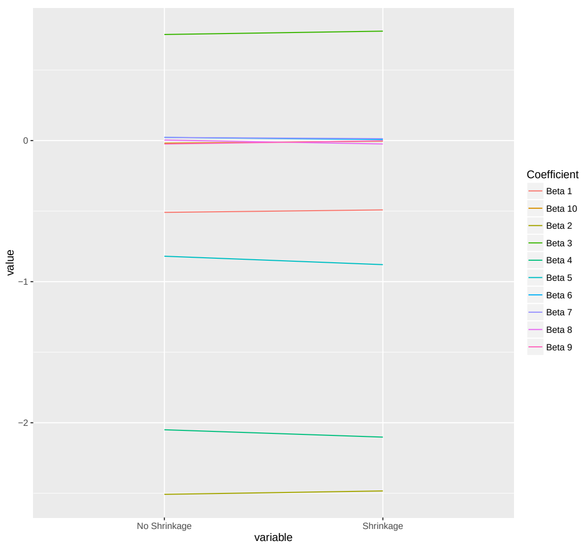


Figure 5: Model (i): $\hat{\beta}$ vs. $\hat{\beta}_{\text{shrink}}$

5.2.2 Results of simulations from Model (ii)

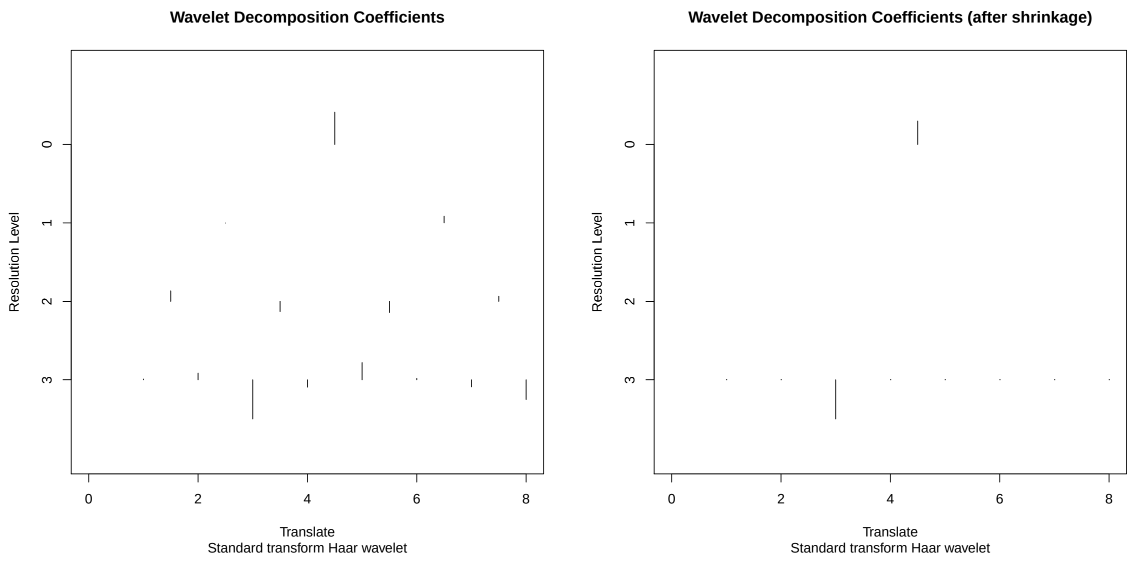


Figure 6: Model (ii): Before and after shrinkage multi-resolution plots of $\hat{\beta}_1$

If the effect size 0.01 is adopted as the threshold of variable inclusion, explanatory variables 1 - 5 are selected, which is consistent with the data generating scheme. Although this method empirically selects the true signals according to certain thresholds, $\hat{\beta}_{\text{shrink}}$, which depends on the wavelet shrinkage algorithm, is not necessarily an accurate estimate of β. For example,using the above two data generating scheme (i.e. Model (i) and Model (ii)), $\hat{\beta}$, $\hat{\beta}_{\text{shrink}}$ and β for some predictor are 2.7811, 0.5065 and 2.7325 respectively. However, such inconsistency can be ameliorated by refitting the linear regression model with the selected predictors.

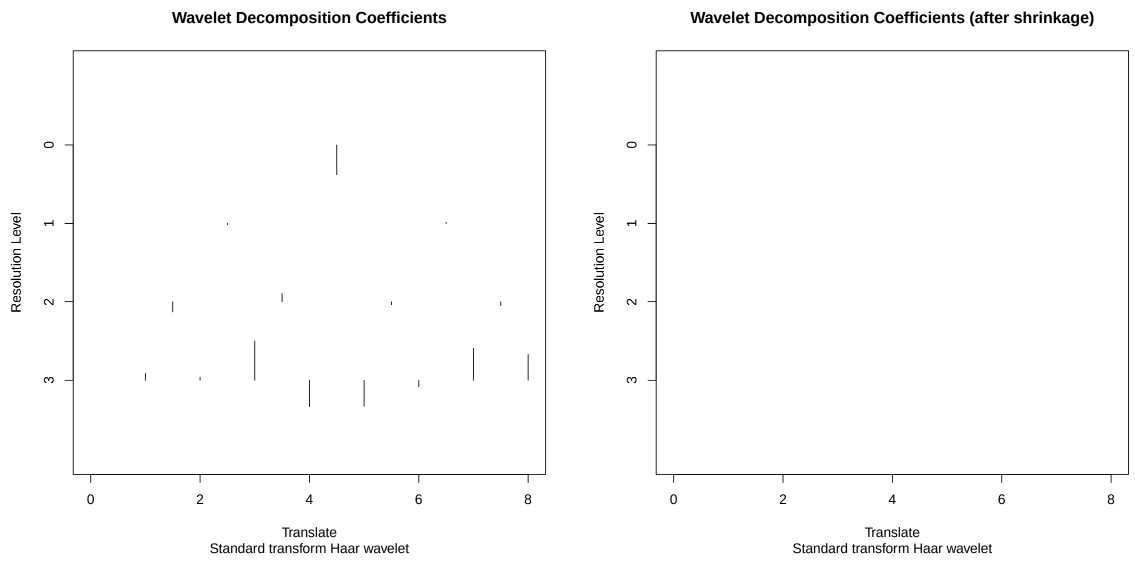


Figure 7: Model (ii):Before and after shrinkage multi-resolution plots of $\hat{\beta}_2$

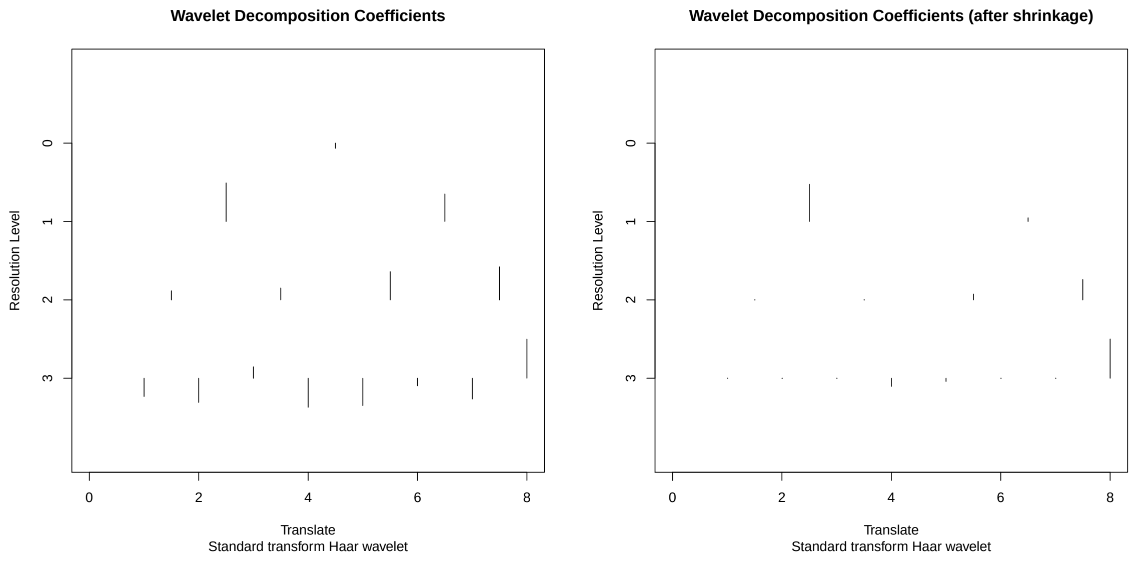


Figure 8: Model (ii):Before and after shrinkage multi-resolution plots of $\hat{\beta}_6$

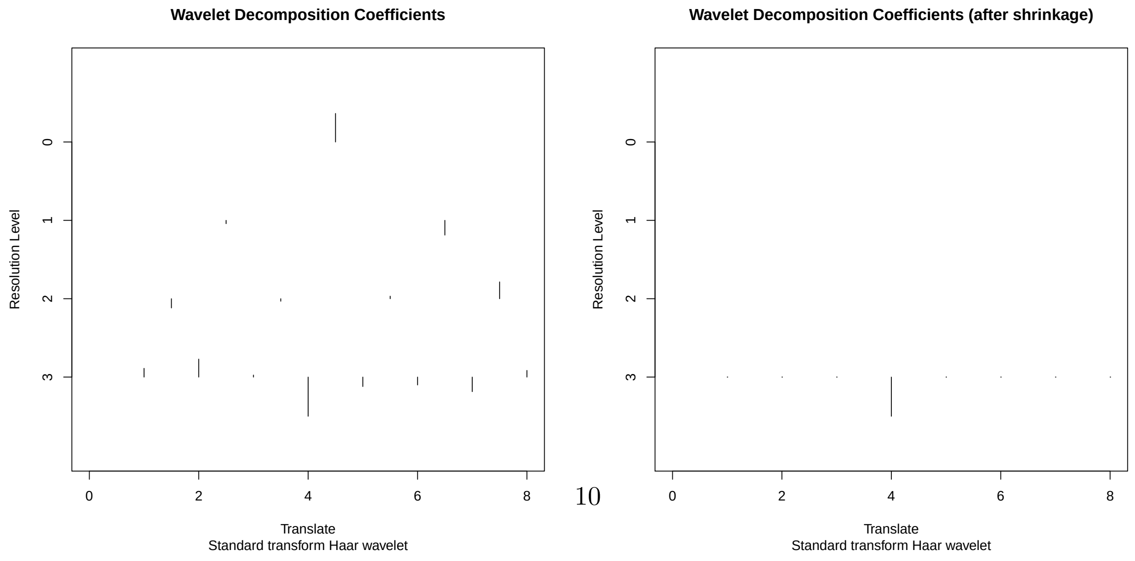


Figure 9: Model (ii):Before and after shrinkage multi-resolution plots of $\hat{\beta}_8$

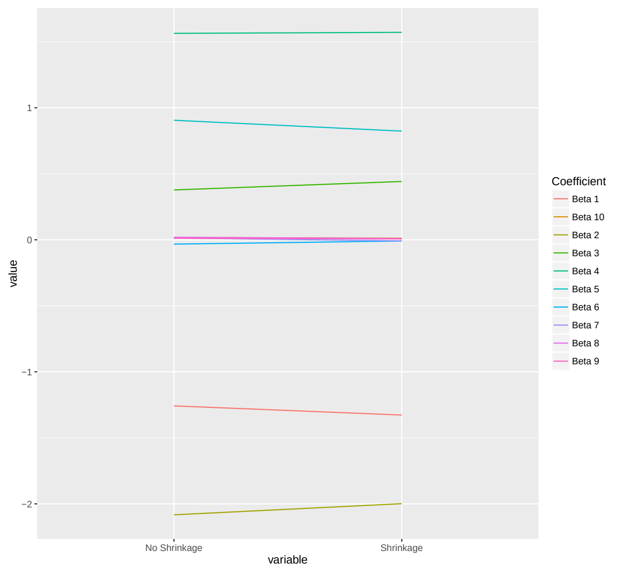


Figure 10: Model (ii): $\hat{\beta}$ vs. $\hat{\beta}_{\text{shrink}}$

6 Concluding remarks

In this paper, we have proposed a parallel framework MapReduce for distributed implementation of wavelet-based linear regression. This new algorithm is shown to outperform the conventional one in terms of computation time when the number of time measurements T is large.

On one hand, although T is assumed to be a power of two in this paper, DWT can be extended to deal with the cases where this condition fails, which is indeed more common. On the other hand, beyond implementation for wavelet-based linear regression, this algorithm is so applicable to regression problems involving orthogonal polynomial coding or other orthogonal transforms. The requirement that the transform be orthogonal allows for easy inversion with needing to invert a matrix.